SEQ2SEQ Professor Forcing

Felipe N. Ducau
Center for Data Science
New York University
fnd212@nyu.edu

May 15, 2017

Abstract

During training, RNN encoder-decoder Neural Machine Translation (NMT) algorithms usually perform one-step-ahead word probability estimation by using ground-truth output tokens. We study how adversarial domain adaptation can be used in these systems, via the application of Professor Forcing algorithm, to encourage the dynamics of the translation system at test time, when it has to sample an entire output sequence from the network, to its dynamics during training. We present a simple extension to the original algorithm to be adapted for the case of producing variable length outputs and show that this approach can effectively improve the BLEU score of the generated translations.

1 Introduction

Recurrent Neural Networks [Graves, 2012] have become the main architecture of generative models for sequential data. One of the applications of these neural networks is for the task of Neural Machine Translation [Cho et al., 2014]. Despite being a relatively new technique, Neural Machine Translation (NMT) has already shown promising results, achieving state-of-the-art performance in various settings. Even though the performance is comparable to, and sometimes better than the conventional Statistical Machine Translation approach [Koehn et al., 2003], there are still certain drawbacks at the moment of generating the target sentence.

One of the problems of the traditional way of training RNNs identified in recent work (Bengio et al., 2015; Huszár, 2015; Li et al., 2016) is that the training procedure is decoupled from the generation procedure. The standard way of training evaluates the likelihood of each token in the sequence given the current state and the previous true token. In the generation (inference) stage, the unknown previous token is replaced by a token generated by the network itself. This disparity between training and generation can yield errors that accumulate along the inference procedure [Bengio et al., 2015]. We will refer to the RNN running with its inputs clamped to the true target sequence as running in *teacher forcing* and when is generating its own input to be *free-running* or closed loop.

Inspired by the Professor Forcing algorithm Lamb et al. [2015], we propose an alternative training procedure and objective functions for sequence to sequence NMT with RNNs based in the Generative Adversarial Learning framework [Goodfellow et al., 2014]. As in Professor Forcing we introduce a new RNN network (adversarial or discriminator network) which is trained to distinguish whether the generative model is running in teacher forcing mode or in free-running mode.

Our approach goes one step forward in the application of Professor Forcing in two main directions: (1) we apply the discriminator network to a variable length sequence of hidden states instead of using fixed size sequences; and (2) we explore how to integrate this idea into a SEQ2SEQ Neural Machine Translation model with attention mechanism [Bahdanau et al., 2014].

2 Related Work

Several adversarial learning approaches have been recently introduced both for NMT and Neural Dialogue Generation tasks. Li et al. [2017] takes an adversarial reinforcement learning approach where the discriminator network operates on the generated sequences and provides feedback to the generator through their Advesrarial REINFORCE algorithm. Bahdanau et al. [2016] proposes an actor critic method from reinforcement learning (RL) in NMT that optimizes directly for BLEU score instead of maximizing the likelihood of the generated sequence.

Another adversarial technique for NMT was recently proposed in Wu et al. [2017] showing some improvements in BLEU scores by training a discriminator to distinguish between real sequences of tokens from generated ones, via Monte-Carlo sampling and the REINFORCE algorithm.

Our work makes use of adversarial training, but differs from these previous approaches in the fact that the discriminator network we use does not take as input the translated sentence, but the hidden states of the generative network instead. Professor Forcing uses adversarial domain adaptation (Ajakan et al., 2014; Ganin et al., 2015) along with the Generative Adversarial Networks [Goodfellow et al., 2014] framework to encourage the dynamics of a Recurrent Neural Network (RNN) during generation to be the same as when training. It was shown in Lamb et al. [2015] that this approach yields qualitatively better samples when the number of generation steps is large. It is worth noticing that, even though the architecture is similar to generative adversarial networks (GANs) because of the use of a discriminator classifier, Professor Forcing tries to determine if a given set of hidden states comes from sampling mode or teacher forcing mode, while GAN classifier looks at the generated samples and determines if it is a real or generated sample. Such a strategy makes the system differentiable avoiding the need to generate gradients using RL techniques.

3 Proposed Approach: SEQ2SEQ Professor Forcing

3.1 Preliminaries

Attention Based RNN Encoder-Decoder

The RNN Encoder-Decoder or SEQ2SEQ model architecture consists of two RNNs: (1) one which encodes a (possibly variable) sequence of symbols into a fixed-length continuous vector representation; and (2) a second one that decodes this representation into another sequence of symbols.

The encoder RNN reads one symbol of the source sequence sequentially and updates its hidden state. When it finish processing the input sequence, the hidden state of the encoder $\mathbf{h}_e$ contains a summary of the input sequence. The decoder network is trained to estimate the probability of the output sequence by predicting sequentially the next symbol y_t given its hidden state $\mathbf{h}_{d\langle t \rangle}$.

For neural machine translation the encoder receives a sentence in the source language and is trained to produce the translation of that sentence in the target language as the output of the decoder network. Generally this model is trained by performing one step ahead word probability estimation to find the parameters θ such that the conditional log likelihood is maximized,

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_{\theta}(y_n | x_n, y_{<n}) \tag{1}$$

This training procedure is also known as *teacher forcing* [Williams and Zipser, 1989] due to the use of ground truth samples y_t being fed back to the model to be conditioned on for the prediction of later outputs. When this network is used at test time, when the ground-truth output sequence is not available, we sample, at each time step, from the predicted conditional probability distribution given the previous samples.

The use of attention for sequence to sequence learning [Bahdanau et al., 2014] further improves the performance of this model in NMT by allowing the discriminator network to use the hidden states of the encoder RNN as an extra input at each time step to compute its current hidden state. In the decoding stage, the decoder chooses to which parts of the source sentence to pay attention to. This alleviates the encoding task of building a single representation of the source sentence into a fixed-length vector. In the case of translation this helps with the word alignment between the source and target sentences.

Professor Forcing

Professor Forcing model, as defined in Lamb et al. [2015], consists on a generator RNN which is trained to properly match the training data, and also to maintain the behavior of the network during generation indistinguishable whether the network is in teacher forcing mode or whether is in free-running mode (where the inputs are self-generated). By using the GAN framework [Goodfellow et al., 2014], an adversarial term can be added to the cost function of the generator RNN to attempt to enforce this two distributions over sequences to match.

3.2 Definitions and Notation

In our case, instead of a single RNN for the generator network, we have a encoder-decoder architecture. We add, on top of the encoder-decoder that performs the translation *per-se* a discriminator network whose role is to inspect the dynamics of the decoder network and try to guess if those dynamics were produced by the decoder working in teacher forcing mode (true samples) or in free running mode (fake samples). The discriminator is a new RNN model that can process variable length inputs.

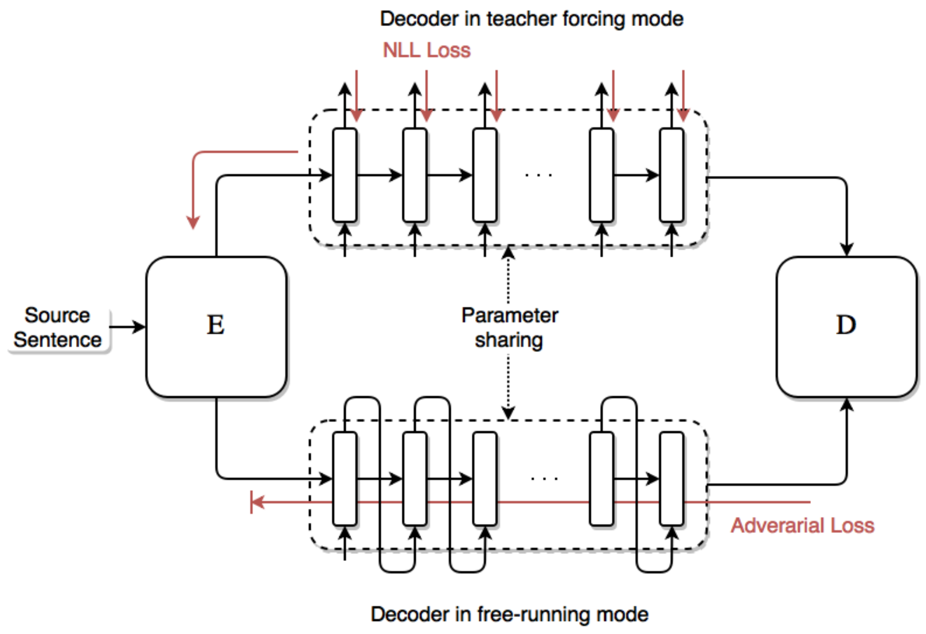

Figure 1: Schematic architecture of the proposed model. An input sequence is encoded by the encoder (E) network and then decoded both in free-running (open loop) which consists in predicting the entire output and in teacher forcing mode (closed loop) performing one step ahead word prediction. A discriminator network is used to classify the dynamics of the decoder network in open or closed loop.

Assume access to a training distribution with pairs of sentences $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is in our source language, while $\mathbf{y}$ is the same sentence in the target language. The bidirectional RNN used as the encoder of our sequence to sequence model receives $\mathbf{x}$, processes it and generates a sequence of hidden states $\mathbf{h}$ composed by the concatenation of the hidden states produced by encoding the input sequence from left to right and from right to left, i.e. $\mathbf{h} = [\overrightarrow{\mathbf{h}}, \overleftarrow{\mathbf{h}}]$. A sequence $\hat{\mathbf{y}}$ is then produced by the decoder network according to the sequence to sequence model distribution $P_\theta(\hat{\mathbf{y}}|\mathbf{x})$. Let θ_e be the parameters of the encoder (sub)network, and θ_g be the parameters of the decoder part of the network (generative part).

The discriminator network D with parameters ϕ_d is a probabilistic classifier that takes as input a behavior sequence $\mathbf{b}$ which represents the dynamics of the decoder network while generating the output sequence $\hat{\mathbf{y}}$ conditioned on the input sequence $\mathbf{x}$ either in teacher forcing or in free-running mode. The behavior sequence is the output of a behavior function $B(\mathbf{x}, \mathbf{y}, \theta_g)$ and will be denoted as $\mathbf{b}_{TF}$ when the decoder is set to work in teacher forcing mode and $\mathbf{b}_{FR}$ when it is running in free-running mode. The discriminator output $D(\mathbf{b})$ estimates the probability that $\mathbf{b}$ was produced in teacher forcing mode.

3.3 Training Objective

The discriminator is trained, as expected, to maximize the likelihood of properly classifying a behavior sequence.

$$\begin{aligned} C_d(\phi_d|\theta_g) =& \mathbb{E}_{(x,y)\sim\text{data}}\big[-\log D(B(\mathbf{x}, \mathbf{y}, \theta_e, \theta_g)) \\ &+ \mathbb{E}_{\mathbf{y}\sim P_{\theta_g}(\mathbf{y}|\mathbf{x})}[-\log(1 - D(B(\mathbf{x}, \mathbf{y}, \theta_g), \theta_d)]\big] \end{aligned} \tag{2}$$

The discriminator parameters are trained with minibatches of N samples of sequences that come from the generator in teacher-forcing mode and N samples of behavior sequences from the discriminator set to free-running mode (i.e. $\mathbf{y}$ distributed according to from $P_{\theta_g}(\mathbf{y}|\mathbf{x})$). Note that the cost function of the discriminator is conditioned on the parameters of the generator: when the generator parameters are updated, the task of the discriminator changes accordingly.

The first part of the network, namely the encoder-decoder architecture parameters are trained trained by means of two different losses: (a) the standard negative log-likelihood to maximize the likelihood of the training data; and (b) an adversarial loss which tries to make the free-running behavior of the network to match the behavior it has when it runs in closed loop. Given that the second loss is closely related with the decoder part of our SEQ2SEQ model, it is designed to only have influence (get back-propagated through) this piece. The reason why we decide to model the backward path of the training procedure this way is because, if we change the encoder part of the network according to the adversarial loss we would be also affecting the teacher forcing behavior by modifying the way it needs to understand its input. Two alternatives for the backpropagation procedure are studied, one in which we update the weights of the attention mechanism according to the adversarial loss, and one in which the discriminator does not affect the attention mechanism.

The cost function of the encoder is then defined as,

$$C_e(\theta_e) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\text{data}}[-\log P_{\theta_e,\theta_g}(\mathbf{y}, \mathbf{x})] \tag{3}$$

and for the generator (decoder) we have,

$$\begin{aligned}C_g(\theta_g) &= C_{g_{NLL}}(\theta_g) + \lambda \cdot C_{g_{Adv}}(\theta_g)\\ &= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\text{data}}[-\log P_{\theta_e,\theta_g}(\mathbf{y},\mathbf{x})] + \lambda \cdot \mathbb{E}_{\mathbf{x}\sim\text{data},\mathbf{y}\sim P_{\theta_g}(\mathbf{y}|\mathbf{x})}[-\log D(B(\mathbf{x},\mathbf{y},\theta_g),\theta_d)]\end{aligned} \tag{4}$$

where λ is an hyperparameter of the network which indicates how much the update should focus on minimizing the adversarial cost with respect to the NLL cost.

In practice we first compute the NLL loss incurred while decoding the sequence in teacher forcing mode and obtain the corresponding gradients w.r.t. both the parameters of the encoder and decoder networks. Then we generate an output sequence using the decoder in free-running mode, compute the behavior sequence $\mathbf{b}$ and $D(\mathbf{b})$ to compute the adversarial loss $C_{g_{Adv}}$ and compute the gradients of this loss with respect to the parameters of the decoder (θ_g). We add this gradients to the corresponding ones from the NLL loss and run backpropagation.

We also explore an alternative adversarial loss. We use the Boundary Seeking GAN (BGAN) cost function for the generator as defined in Devon Hjelm et al. [2017]. When we use this cost function, the generator cost will be

$$\begin{aligned}C_g(\theta_g) &= C_{g_{NLL}}(\theta_g) + \lambda \cdot C_{g_{Adv\_BGAN}}(\theta_g)\\ C_g(\theta_g) &= C_{g_{NLL}}(\theta_g) + \lambda \cdot \mathbb{E}_{\mathbf{x}\sim\text{data},\mathbf{y}\sim P_{\theta_g}(\mathbf{y}|\mathbf{x})}\left[\big(\log D(B(\mathbf{x},\mathbf{y},\theta_g),\phi_d) - \log(1 - D(B(\mathbf{x},\mathbf{y},\theta_g),\phi_d))\big)^2\right]\end{aligned} \tag{5}$$

3.4 Discriminator Network

Given that the input of the discriminator is naturally a variable length sequence, we define the discriminator to be a bidirectional RNN. We explore two versions of such discriminator, both illustrated in Figure 2: in the first version, (1) which we will refer as *simple* we encode the behavior sequence $\mathbf{b}$ with a bidirectional RNN and feed the two last hidden states to a multi-layer perceptron (MLP) network followed by a sigmoid non-linearity to obtain the estimation of the probability that $\mathbf{b}$ was produced by the generator in open loop; the second approach (2) is to individually run the respective pairs of hidden states of the discriminator through a MLP network and obtain that probability step-wise. We noticed no significant difference between the two approaches throughout our experiments.

4 Experiments

4.1 Experimental Setup

4.1.1 Dataset

We train our models in German to English (Ge→En) translation. The training and validation datasets used are random subsets of WMT 2015 consisting on 1,751,500

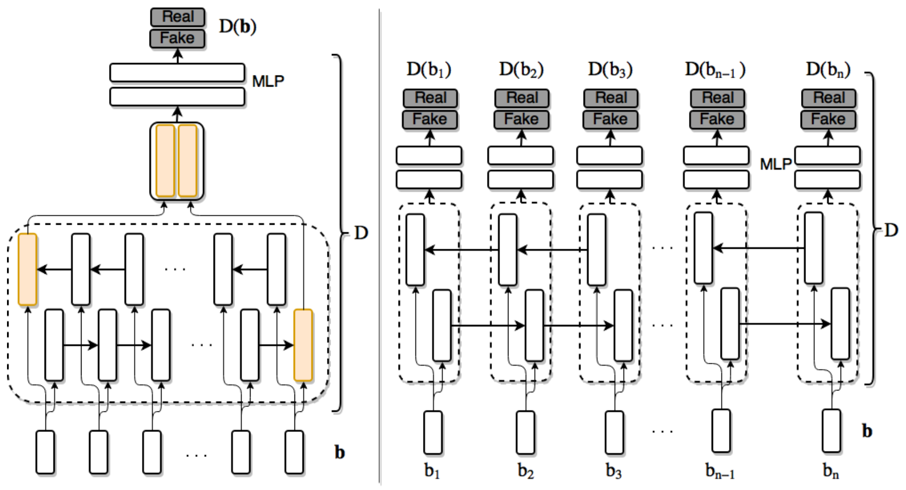

Figure 2: Discriminator network. It takes as input a behavior sequence $\mathbf{b}$ and estimates the probability of being generated by the decoder in teacher-forcing mode. The network in the *left* is referred as *simple* discriminator and computes a single estimation for this probability. In the *right* we present the *complete* discriminator network which emits one probability estimation for each element of the behavior sequence $\mathbf{b}$.

pairs of sentences for training and 2170 pairs of sentences for validation. The test set is a random subset of WMT 2016 containing 3000 pairs of sentences. The maximal sentence length considered for training was set to 50 for all our experiments although for validation and test we do not impose any constraints on the sentence length. We keep the top 20k words in the training set and set the remaining ones to the special *UNK* token.

4.1.2 Implementation

All the RNNs used in our architecture have a single hidden layer of gated recurrent units (GRU), following the design in Cho et al. [2014]. For all the experiments the words are embedded in a continuous space of 620 dimensions, the hidden states of the recurrent units are of size 1000. In the reported results we did not use dropout since it did not show an improvement in the final computed BLEU scores in side experiments.

The MLP consists of two linear layers of 2000 hidden units each followed by a ReLU non-linearity. The output layer composes an affine transformation and a sigmoid unit to output a single integer. The behavior function B, as in Lamb et al. [2015], outputs the pre-tanh activation of the GRU hidden states.

All the code for the experiment was written using Theano framework [Al-Rfou et al., 2016] and is available in the GitHub repository of this project [1].

[1]`https://github.com/fducau/nmt`

4.2 Training Procedure

The entire network is trained end to end by minibatch stochastic gradient descent with adaptive learning rate, following the Adadelta method [Zeiler, 2012] using a learning rate of 0.01. We trained with minibatches of 16 sentences. Gradient clipping was used to avoid exploding gradients in the RNN. Early stopping on BLEU score was used to determine the end of the training phase.

When the discriminator is too poor in recognizing free-running from teacher forced sequences, the feedback it provides to the generator in form of gradients can be harmful. On the other end, if the discriminator is very good at telling apart these sequences, it will make the generator not to learn properly. As suggested in Lamb et al. [2015] we use an heuristic way of balancing the training procedure of these two networks. At every minibatch of decoded samples (N in teacher forcing and N in free-running) we evaluate the accuracy on the discriminator. If the accuracy is lower than 75% we update only the discriminator and not the generator, when it is higher than 95% we update the parameters of the generator and not the ones of the discriminator. Lastly if it is within those bounds, both network update their parameters.

Algorithm 1 illustrates the training procedure followed to train this model.

Algorithm 1: Minibatch stochastic gradient descent for the adversarial model.

```
Data: Pairs of training sentences $(\mathbf{x}, \mathbf{y})$, $\lambda$
for number of training steps do
    Sample $(\mathbf{x}, \mathbf{y})$ from data ;
    Estimate $P_{\theta_g}(\mathbf{y}|\mathbf{x})$ by running the encoder-decoder in teacher forcing mode;
    Get $\mathbf{b}_{TF}$ from the hidden states of the decoder in teacher forcing mode;
    Compute NLL losses $C_e$ and $C_{g_{NLL}}$ ;
    $C_g \leftarrow C_{g_{NLL}}$ Sample $\hat{\mathbf{y}}_{FR} \sim P_{\theta_g}$ from encoder-decoder in free-running
      mode;
    Get $\mathbf{b}_{FR}$ from the hidden states of the decoder in free-running mode;
    Compute the discriminator predictions $D(\mathbf{b}_{TF})$ and $D(\mathbf{b}_{FR})$ ;
    Compute discriminator accuracy $\alpha_D$ ;
    if $\alpha_D < 0.95$ then
        Update discriminator weights by following the stochastic gradient on the
        cost $C_d$;
    end
    if $\alpha_D > 0.75$ then
        Compute discriminator predictions $D(\mathbf{b}_{FR})$ ;
        Compute adversarial loss $C_{g_{Adv}}$ ;
        $C_g \leftarrow C_g + \lambda \cdot C_{g_{Adv}}$ ;
    end
    Update encoder weights by following the stochastic gradient on the cost $C_e$;
    Update decoder weights by following the stochastic gradient on the cost $C_g$;
end
```

4.3 Results

Based in the previous subsections we define the following experimental setups:

- **Baseline**: RNN Encoder Decoder.
- **Adv-L[x]**: SEQ2SEQ Professor Forcing with $\lambda = x$
- **Adv-L[x]-Att**: SEQ2SEQ Professor Forcing with $\lambda = x$ and using the discriminator feedback to update the weights of the attention mechanism as well as the rest of the generator network.
- **BGAN-L[x]**: SEQ2SEQ Professor Forcing with $\lambda = x$ and using the discriminator cost function from equation 5

The results of the experiments are summarized in table 4.3. Figure 3 shows the validation curves during training. The first observation is that even though there is small difference in the validation NLL for all the experiments, there is almost one point of improvement in BLEU score in the test set when using the *Adv-L15* architecture. This behavior seems to indicate that the adversarial approach is effectively improving the generation procedure when the decoder of the network is in free-running mode, without affecting the optimization algorithm used to minimize the NLL.

Another thing to note is that the convergence of the optimization algorithm is not affected by adding the adversarial network (in terms of iterations). The implementation actually takes about 1.7 times more per epoch when training in Professor Forcing than in teacher forcing.

We also evaluated the performance of the models exclusively in long sentences (more than 50 words length as used in training), which is the setting in which Lamb et al. [2015] obtains better results compared with teacher forcing, but found no further improvements.

Model	Valid NLL	Test BLEU (beam-search $k = 5$)	Test BLEU (Greedy)
Baseline	63.59	15.94	13.95
Adv-L1	63.45	16.34	14.85
Adv-L15	63.86	**16.85**	**15.23**
Adv-L150	64.19	16.46	14.16
Adv-L150-Att	63.82	15.91	14.49
BGAN-L1	63.65	15.79	14.93

Table 1: Validation NLL and test BLEU scores for the defined experiments.

5 Conclusion and Future Work

In this work we evaluated the effect of adding an adversarial network to a RNN encoder-decoder architecture with the objective of matching the dynamics of the decoder network when running in test mode vs when its inputs are forced to be the ones of the real target sentence. We introduced a lightweight way of introducing the concepts of

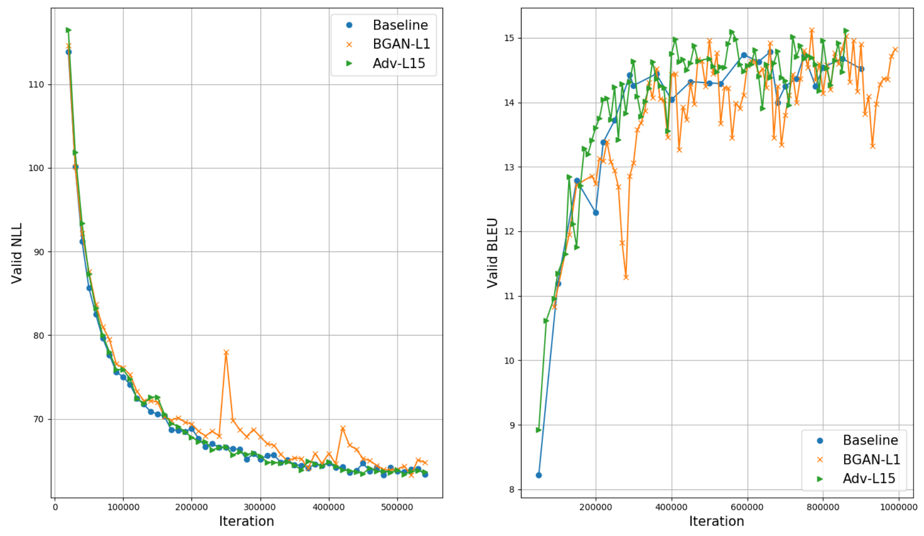

Figure 3: Validation NLL and BLEU scores during training for three of the proposed experiments.

Professor Forcing in a complex architecture by only backpropagating the loss through the decoder part of the SEQ2SEQ model. We found that the addition of an extra network does not affect the optimization procedure of minimizing the negative log likelihood loss and at the same time improves the performance of the architecture in test mode.

We also acknowledge that further experimentation is necessary to further explore the capabilities of our approach. Little hyperparameter tuning was performed, being the first thing to try to improve the presented results. Another topic to be explored is how this method works with different regularization techniques for RNNs. Yet other interesting approach to explore is the use of sampling when running the decoder network in free-running mode during training instead of using the token with the maximum probability. Lastly we would like to further explore extensions to the behavior function, mainly adding the attention weights at each time step.

6 Acknowledgements

We would like to thank especially Stanislas Lauly whose guidance was a fundamental for the development of this project. We also want to thank Martín Arjovsky and Jumbo Jake Zhao for their helpful feedback.

References

Hana Ajakan, Pascal Germain, Hugo Larochelle, Franois Laviolette, and Mario Marchand. Domain-adversarial neural networks. 2014. URL `http://arxiv.org/`

abs/1412.4446.

Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, and et al. Theano: A python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688, 2016. URL http://arxiv.org/abs/1605.02688.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL http://arxiv.org/abs/1409.0473.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. 2016. URL http://arxiv.org/abs/1607.07086.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *ArXiv e-prints*, June 2015.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

R Devon Hjelm, A. P. Jacob, T. Che, K. Cho, and Y. Bengio. Boundary-Seeking Generative Adversarial Networks. *ArXiv e-prints*, February 2017.

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-Adversarial Training of Neural Networks. *ArXiv e-prints*, May 2015.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. 2014. URL http://arxiv.org/abs/1406.2661.

Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012. ISBN 978-3-642-24796-5. doi: 10.1007/978-3-642-24797-2. URL http://dx.doi.org/10.1007/978-3-642-24797-2.

F. Huszár. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *ArXiv e-prints*, November 2015.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073462. URL http://dx.doi.org/10.3115/1073445.1073462.

Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *arXiv:1610.09038 [cs, stat]*, 2015. URL `http://arxiv.org/abs/1610.09038`.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. 2016. URL `http://arxiv.org/abs/1606.01541`.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. 2017. URL `http://arxiv.org/abs/1701.06547`.

R. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. Adversarial neural machine translation. 2017. URL `http://arxiv.org/abs/1704.06933`.

Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL `http://arxiv.org/abs/1212.5701`.